



General Inner Approximation of Vector-valued Functions

Olivier Mullier, Eric Goubault, Michel Kieffer, Sylvie Putot

► To cite this version:

Olivier Mullier, Eric Goubault, Michel Kieffer, Sylvie Putot. General Inner Approximation of Vector-valued Functions. *Reliable Computing*, 2013, 18, pp.117-143. hal-00935773

HAL Id: hal-00935773

<https://hal-centralesupelec.archives-ouvertes.fr/hal-00935773>

Submitted on 24 Jan 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

General Inner Approximation of Vector-valued Functions

Olivier MULLIER

CEA Saclay Nano-Innov; Institut CARNOT CEA
LIST DILS/MeASI; Point Courrier n°174 91191 Gif-
sur-Yvette Cedex, France

`Olivier.Mullier@cea.fr`

Éric GOUBAULT

CEA Saclay Nano-Innov; Institut CARNOT CEA
LIST DILS/MeASI; Point Courrier n°174 91191 Gif-
sur-Yvette Cedex, France

`Eric.Goubault@cea.fr`

Michel KIEFFER

Laboratoire des Signaux et Systèmes (L2S) - CNRS -
Supelec - Univ Paris-Sud 91192 Gif-sur-Yvette LTCI -
CNRS - Telecom ParisTech 75013 Paris Institut uni-
versitaire de France

`Michel.Kieffer@lss.supelec.fr`

Sylvie PUTOT

CEA Saclay Nano-Innov; Institut CARNOT CEA
LIST DILS/MeASI; Point Courrier n°174 91191 Gif-
sur-Yvette Cedex, France

`Sylvie.Putot@cea.fr`

November 15, 2013

Abstract

This paper addresses the problem of evaluating a subset of the range of a vector-valued function. It is based on a work by Goldsztejn and Jaulin which provides methods based on interval analysis to address this problem when the dimension of the domain and co-domain of the function are equal. This paper extends this result to vector-valued functions with domain and co-domain of different dimensions. This extension requires the knowledge of the rank of the Jacobian function on the whole domain. This leads to the sub-problem of extracting an interval sub-matrix of maximum rank from a given interval matrix. Three different techniques leading to approximate solutions of this extraction are proposed and compared.

1 Introduction

Computing the values a function can take over some domain is generally of great interest in the analysis of numerical programs as in abstract interpretation [3], in robust control of dynamic systems [12], or in global optimization [19]. Computing the image of a domain by a function (also called direct image or *range*) exactly is intractable in general. The classical solution is then to compute an outer approximation of this range, which can unfortunately be very pessimistic. This outer approximation may introduce many values which do not belong to the range. Providing, in addition, an inner approximation can be helpful to state the quality of the outer approximation.

For scalar-valued functions, an inner approximation can be evaluated using modal intervals [7] (using Kaucher arithmetic [13]) or twin arithmetic [17]. When f maps \mathbb{R}^n to \mathbb{R}^n , modal intervals [4] can also be used in the linear case. For the non-linear case, set inversion [11] can be used when f is *globally* invertible (when an inverse function f^{-1} can be produced).

In the more general case of f being *locally* invertible, the method described by Goldsztejn and Jaulin in [5] can be applied. This technique requires however the inverse of the Jacobian of f . Thus it can only be applied for functions from \mathbb{R}^n to \mathbb{R}^n of constant rank n .

This paper proposes a generalization of the method in [5] to deal with functions f from $\mathcal{D} \subseteq \mathbb{R}^m$ to \mathbb{R}^n , with $m \neq n$, with rank r . It describes a method to compute an inner approximation for at most r components of f . As in [5], the evaluation of the Jacobian of the function on a given subset of its domain is needed. There, the identification of the components that can be used to compute an inner approximation has to be done by extracting the sub-matrix of full rank in its Jacobian. Checking regularity of interval matrices is a NP-hard problem [20], so is the problem of extracting an interval sub-matrix of full rank. To our knowledge, no necessary and sufficient condition for checking regularity can be used to address this problem (a list of necessary and sufficient conditions for an interval matrix to be regular can be found in [21]).

Concurrently with our work, Ishill et al. [10] proposed a computation of a projection of the solution set of under-constrained systems. This is closely related to our problem, since the image of a function can be recast in terms of constraint solving. But their method relies on the classical Hansen Sengupta operator, while we modify it to handle the different cases according to the dimension of domain, codomain and the rank of the function. This restricts their method to the case where the rank of the function is equal to the dimension of the codomain, or to a manual choice of some dimensions.

This paper is organized as follows: Section 2 recalls the main result of [5] on the computation of an inner approximation of the range of vector-valued functions with domain and co-domain of the same dimension. Section 3 describes the extension of this result to functions with domain and co-domain of different dimensions. Section 4 addresses algorithms for computing and inner approximation and describes how sub-matrices of full rank can be extracted from a given interval matrix using different techniques. The computation of an inner

approximation of the range of functions is illustrated on examples in Section 5.

Notations

$\mathbf{x} = [\underline{x}, \bar{x}] \triangleq \{x \in \mathbb{R} : \underline{x} \leq x \leq \bar{x}\}$ is an *interval* where \underline{x} and \bar{x} are respectively its lower and its upper bound. $\mathbb{IR} \triangleq \{[\underline{x}, \bar{x}] : \underline{x}, \bar{x} \in \mathbb{R}, \underline{x} \leq \bar{x}\}$ represents the set of intervals. A box is the Cartesian product of n intervals in \mathbb{IR}^n . For an interval $\mathbf{x} = [\underline{x}, \bar{x}]$, the *width* is $\text{wid}(\mathbf{x}) \triangleq \bar{x} - \underline{x}$, the *midpoint* is $\text{mid}(\mathbf{x}) \triangleq \frac{1}{2}(\bar{x} + \underline{x})$, the *interior* is $\text{int}(\mathbf{x}) \triangleq \{x \in \mathbb{R} | \underline{x} < x < \bar{x}\}$, and the *boundary* is denoted by $\partial\mathbf{x}$. The *magnitude* is denoted $|\mathbf{x}| \triangleq \max\{|\underline{x}|, |\bar{x}|\}$ and the *mignitude* is $\langle \mathbf{x} \rangle \triangleq \min\{|\underline{x}|, |\bar{x}|\}$ if $0 \notin \mathbf{x}$ and $\langle \mathbf{x} \rangle = 0$ otherwise. The width of an interval vector $\mathbf{x} \in \mathbb{IR}^n$ is $\max_{1 \leq i \leq n}(\text{wid}(\mathbf{x}_i))$.

The core of interval analysis is its fundamental theorem (see, *e.g.*, [16] or [18]) asserting that an evaluation of an expression using intervals gives an outer approximation of the range of this expression over the considered intervals. An interval function is an inclusion function denoted here \mathbf{f} : $f(\mathbf{x}) = \{f(x) : x \in \mathbf{x}\} \subseteq \mathbf{f}(\mathbf{x})$ for \mathbf{x} included in the domain of f . For an interval square matrix $\mathbf{A} \in \mathbb{IR}^{n \times n}$, $\text{Diag } \mathbf{A} \in \mathbb{IR}^{n \times n}$ is the diagonal interval matrix whose diagonal entries are $(\text{Diag } \mathbf{A})_{ii} = \mathbf{A}_{ii}$, $1 \leq i \leq n$, and 0 elsewhere. $\text{OffDiag } \mathbf{A} \in \mathbb{IR}^{n \times n}$ is the interval matrix with null diagonal and with off-diagonal entries such that $(\text{OffDiag } \mathbf{A})_{ij} = \mathbf{A}_{ij}$. For a vector-valued function $f : \mathcal{D} \subseteq \mathbb{R}^m \rightarrow \mathbb{R}^n$ and $x \in \mathcal{D}$, $f_{i:j}(x) \triangleq (f_i(x), f_{i+1}(x), \dots, f_j(x))^T$ for $i \leq j$. For the Jacobian J^f of f and $x \in \mathcal{D}$,

$$J^{f_{i:j}, x_{k:\ell}}(x) \triangleq \begin{pmatrix} \frac{\partial f_i}{\partial x_k}(x) & \frac{\partial f_i}{\partial x_{k+1}}(x) & \dots & \frac{\partial f_i}{\partial x_\ell}(x) \\ \frac{\partial f_{i+1}}{\partial x_k}(x) & \frac{\partial f_{i+1}}{\partial x_{k+1}}(x) & \dots & \frac{\partial f_{i+1}}{\partial x_\ell}(x) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_j}{\partial x_k}(x) & \frac{\partial f_j}{\partial x_{k+1}}(x) & \dots & \frac{\partial f_j}{\partial x_\ell}(x) \end{pmatrix}$$

is the restriction of the Jacobian of f for $j - i + 1$ components of f and $\ell - k + 1$ components of x . $I_k \in \mathbb{R}^{k \times k}$ is the identity matrix of dimension k . The null matrix with k rows and ℓ columns is denoted $0_{k \times \ell}$ and the null vector of k entries is denoted $0_k \triangleq (0, \dots, 0)^T$.

2 Inner approximation for functions with domain and co-domain of the same dimension

This section recalls the main result of [5] to evaluate an inner approximation of the range of a function with domain and co-domain of the same dimension.

Corollary 2.1 *Let $\mathbf{x} \in \mathbb{IR}^n$ and $f : \mathbf{x} \rightarrow \mathbb{R}^n$ be a continuous function continuously differentiable in $\text{int}(\mathbf{x})$. Consider $\mathbf{y} \in \mathbb{IR}^n$ and $\hat{x} \in \mathbf{x}$ such that $f(\hat{x}) \in \mathbf{y}$. Consider also an interval matrix $\mathbf{J} \in \mathbb{IR}^{n \times n}$ such that $f'(x) \in \mathbf{J}$ for all $x \in \mathbf{x}$.*

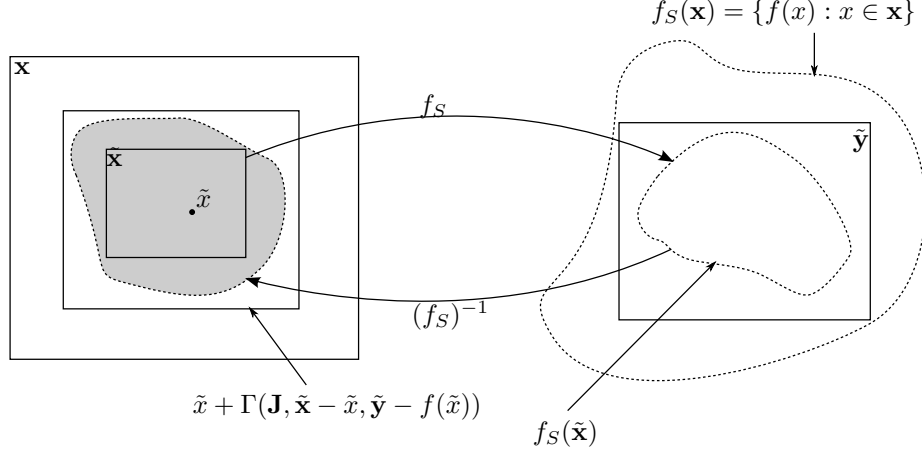


Figure 1: Sets and functions involved in Corollary 2.1 for inner approximation.

Assume that $0 \notin \mathbf{J}_{ii}$ for all $i \in [1, \dots, n]$. Let

$$H(\mathbf{J}, \tilde{x}, \mathbf{x}, \mathbf{y}) = \tilde{x} + (\text{Diag}^{-1} \mathbf{J}) \left(\mathbf{y} - f(\tilde{x}) - (\text{OffDiag} \mathbf{J})(\mathbf{x} - \tilde{x}) \right) \quad (1)$$

If $H(\mathbf{J}, \tilde{x}, \mathbf{x}, \mathbf{y}) \subseteq \text{int}(\mathbf{x})$ then $\mathbf{y} \subseteq \text{range}(f, \mathbf{x})$.

This corollary provides an efficient test for a box \mathbf{y} to be a subset of the range of a vector-valued function. It can be used to compute an inner approximation of functions f from \mathbb{R}^n to \mathbb{R}^n , see Section 4. The restriction on f having same dimension of domain and co-domain comes from the matrix inversion of $\text{Diag} \mathbf{J}$ in (1). Figure 1 illustrates the computation in Corollary 2.1. The left part of Figure 1 represents the domain \mathbf{x} and the right part the co-domain of f . The set-valued map f_S is defined from $\mathcal{P}(\mathbb{R}^n)$ (power set of \mathbb{R}^n) to $\mathcal{P}(\mathbb{R}^n)$ and returns the set $\{f(x) : x \in \mathcal{D}\}$ for a given set \mathcal{D} , see [2]. From a given box $\tilde{\mathbf{x}} \subset \mathbf{x}$, one wants to know if the box $\tilde{\mathbf{y}}$ computed by an inclusion function of f over $\tilde{\mathbf{x}}$ belongs to the range of f or equivalently if $\tilde{\mathbf{y}} = \mathbf{f}(\tilde{\mathbf{x}})$ is a subset of $f_S(\mathbf{x})$. If $\tilde{\mathbf{x}}$ is too large compared to \mathbf{x} , one might have $\tilde{\mathbf{y}} = \mathbf{f}(\tilde{\mathbf{x}}) \not\subseteq f_S(\mathbf{x})$. To prove that $\tilde{\mathbf{y}} \subseteq f_S(\mathbf{x})$, it is sufficient to prove that $f_S^{-1}(\tilde{\mathbf{y}}) \subset \mathbf{x}$. The function $H(\mathbf{J}, \tilde{x}, \mathbf{x}, \mathbf{y})$ in (1) can be seen as an inclusion function for $f_S^{-1} \circ f_S(\mathbf{x})$.

3 Extension for functions with domain and co-domain of different dimensions

Corollary 2.1 only applies for functions having the same dimension for domain and co-domain. It also needs that the determinant of the Jacobian is different from 0. Consider now the case of a function f with domain and co-domain of different dimensions. In what follows, assume that $f : \mathcal{D} \subseteq \mathbb{R}^m \rightarrow \mathbb{R}^n$ is a C^1

function of rank greater than or equal to r on \mathcal{D} . It is assumed that there exist r components $(x_{i_1}, \dots, x_{i_r})$ of $x = (x_1, \dots, x_m) \in \mathcal{D} \subseteq \mathbb{R}^m$ and r components $(f_{j_1}(x), \dots, f_{j_r}(x))$ such that

$$\forall x \in \mathcal{D} \subseteq \mathbb{R}^m, \det \left(\frac{\partial f_{j_k}}{\partial x_{i_\ell}}(x) \right)_{1 \leq k, \ell \leq r} \neq 0 \quad (2)$$

Hereafter, without loss of generality, f is considered after the permutation of the r coordinates $(x_{i_1}, \dots, x_{i_r})$ and the r coordinates $(f_{j_1}(x), \dots, f_{j_r}(x))$ (this permutation is discussed later in Section 4.3). It means that the Jacobian of f has an $r \times r$ sub-matrix on the upper left such that

$$\forall x \in \mathcal{D} \subseteq \mathbb{R}^m, \det \left(\frac{\partial f_j}{\partial x_k}(x) \right)_{1 \leq j, k \leq r} \neq 0 \quad (3)$$

Theorem 3.1 in [5] provides sufficient conditions for a box \mathbf{y} to be included in the range of a function. Theorem 3.1 bellow generalizes this characterization by providing sufficient conditions for a box $\mathbf{y}_1 \in \mathbb{IR}^r$ to be inside the projection on the first r components of the image of $f : \mathcal{D} \subseteq \mathbb{R}^m \rightarrow \mathbb{R}^n$ when f verifies (3).

Theorem 3.1 *Let $f : \mathcal{D} \subseteq \mathbb{R}^m \rightarrow \mathbb{R}^n$ be a C^1 function that verifies (3), $\mathbf{u} \subset \mathcal{D}$ a box in \mathbb{IR}^m and $\mathbf{y}_1 \in \mathbb{IR}^r$. Assume that the two following conditions are satisfied*

- (i) $\mathbf{y}_1 \cap f_{1:r}(\partial \mathbf{u}) = \emptyset$;
- (ii) $f_{1:r}(\tilde{\mathbf{u}}) \in \mathbf{y}_1$ for some $\tilde{\mathbf{u}} \in \mathbf{u}$,

then $\mathbf{y}_1 \subseteq f_{1:r}(\mathbf{u})$.

Before starting the proof the next result is needed.

Lemma 3.1 *Let $f : \mathcal{D} \rightarrow \mathbb{R}^n$ be a C^1 function satisfying (3) and let E be a compact such that $E \subset \mathcal{D}$. Then one has $\partial(f_{1:r}(E)) \subseteq f_{1:r}(\partial E)$.*

Proof: Consider any $y_1 \in \partial(f_{1:r}(E))$. As f is continuous, $f_{1:r}$ is continuous as well. Then the image of E , compact, by $f_{1:r}$ is also compact. In particular it is closed so $\partial(f_{1:r}(E))$ is included in $f_{1:r}(E)$. So there exists $x \in E$ such that $y_1 = f_{1:r}(x)$. Now suppose that $x \in \text{int } E$. We now prove that this leads to a contradiction. As $x \in \text{int } E$, there exists U open of E with $x \in U$. Because of (3), $f_{1:r}$ is a submersion and as submersions are open maps (see [24]), $V = f_{1:r}(U)$ is open in \mathbb{R}^r . We have $y_1 \in V$ then $y_1 \in \text{int } f_{1:r}(E)$ which contradicts $y_1 \in \partial(f_{1:r}(E))$. As a conclusion we have $x \in \partial E$ and eventually, $\partial(f_{1:r}(E)) \subseteq f_{1:r}(\partial E)$. \square *Proof:* [Theorem 3.1] \mathbf{u} is a compact of \mathcal{D} and f is a C^1 function that verifies (3) so we have, from Lemma 3.1, $\partial(f_{1:r}(\mathbf{u})) \subseteq f_{1:r}(\partial \mathbf{u})$. So $\mathbf{y}_1 \cap \partial(f_{1:r}(\mathbf{u})) \subseteq \mathbf{y}_1 \cap f_{1:r}(\partial \mathbf{u}) = \emptyset$ therefore

$$\mathbf{y}_1 \cap \partial(f_{1:r}(\mathbf{u})) = \emptyset. \quad (4)$$

The set $f_{1:r}(\mathbf{u})$ is compact because \mathbf{u} is compact and $f_{1:r}$ is continuous. Let \tilde{u} and $y_1 = f_{1:r}(\tilde{u}) \in \mathbf{y}_1$ be given in (ii). As the intersection of \mathbf{y}_1 and $\partial f_{1:r}(\mathbf{u})$ is empty by (4), $y_1 \in \text{int } f_{1:r}(\mathbf{u})$. Consider any $z \in \mathbf{y}_1$ and suppose that $z \notin f_{1:r}(\mathbf{u})$. Since \mathbf{y}_1 is path connected, there exists a path included in \mathbf{y}_1 between y_1 and z such that, by Lemma A.1. in [5], this path intersects $\partial f_{1:r}(\mathbf{u})$ which is not possible from (4). Therefore $z \in f_{1:r}(\mathbf{u})$ which concludes the proof. \square

Theorem 3.1 is a generalization of Theorem 3.1 in [5] for a function satisfying (3). In Theorem 3.1 in [5], the set $\Sigma = \{x \in \text{int } \mathbf{x} \mid \det f'(x) = 0\}$ can be extended for $f_{1:r}$ by $\Sigma_2 = \{x \in \text{int } \mathbf{x} \mid \text{rank}(f_{1:r}(x)) < r\}$. Due to (3), one has $\Sigma_2 = \emptyset$. In what follows, Corollary 3.1 of Theorem 3.1, which extends the inclusion test of Corollary 2.1, is introduced.

Corollary 3.1 *Let $f : \mathcal{D} \subseteq \mathbb{R}^m \rightarrow \mathbb{R}^n$ be a C^1 function that satisfies (3) and $\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2) \in \mathbb{R}^r \times \mathbb{R}^{m-r}$. Consider $\mathbf{y}_1 \in \mathbb{R}^r$, $\tilde{u} = (\tilde{u}_1, \tilde{u}_2) \in (\mathbf{u}_1, \mathbf{u}_2)$ such that $f_{1:r}(\tilde{u}) \in \mathbf{y}_1$ and $\mathbf{J}^{f_{1:r}} = (\mathbf{J}^{f_{1:r}, u_1} \mathbf{J}^{f_{1:r}, u_2}) \in \mathbb{R}^{r \times m}$ an interval matrix containing $J^{f_{1:r}}$ the Jacobian of $f_{1:r}$ on $(\mathbf{u}_1, \mathbf{u}_2)$ such that $0 \notin (\mathbf{J}^{f_{1:r}, u_1})_{ii}$ for $1 \leq i \leq r$. Let*

$$H_{f_{1:r}}(\mathbf{J}^{f_{1:r}}, \tilde{u}, \mathbf{u}, \mathbf{y}_1) = \tilde{u}_1 + (\text{Diag}^{-1} \mathbf{J}^{f_{1:r}, u_1}) \times \\ \left(\mathbf{y}_1 - f_{1:r}(\tilde{u}) - (\text{OffDiag } \mathbf{J}^{f_{1:r}, u_1})(\mathbf{u}_1 - \tilde{u}_1) - \mathbf{J}^{f_{1:r}, u_2}(\mathbf{u}_2 - \tilde{u}_2) \right).$$

If

$$H_{f_{1:r}}(\mathbf{J}^{f_{1:r}}, \tilde{u}, \mathbf{u}, \mathbf{y}_1) \subseteq \text{int}(\mathbf{u}_1), \quad (5)$$

then

$$\mathbf{y}_1 \subseteq f_{1:r}(\mathbf{u}).$$

Proof: It is sufficient to prove that if (5) is satisfied, the conditions of Theorem 3.1 are satisfied too.

(i) Let $u = (u_1, u_2) \in \partial \mathbf{u}$. Since $u \in \mathbf{u}$, the mean value theorem applied to $f_{1:r}$ (see [18]) shows that

$$f_{1:r}(u) \in f_{1:r}(\tilde{u}) + \mathbf{J}^{f_{1:r}}(\mathbf{u} - \tilde{u}) \quad (6)$$

Let us show that $f_{1:r}(\tilde{u}) + \mathbf{J}^{f_{1:r}}(\mathbf{u} - \tilde{u}) \cap \mathbf{y}_1 \neq \emptyset$ which implies (i) false contradicts (5). Assume that there exists $J \in \mathbf{J}^{f_{1:r}}$, $J = (J_1 J_2)$ with $J_1 \in \mathbb{R}^{r \times r}$ and $J_2 \in \mathbb{R}^{r \times m-r}$; $u = (u_1, u_2)^T$, $\tilde{u} = (\tilde{u}_1, \tilde{u}_2)^T$, and $y_1 \in \mathbf{y}_1$ such that

$$\begin{aligned} y_1 &= f_{1:r}(\tilde{u}) + J(u - \tilde{u}) \\ &= f_{1:r}(\tilde{u}) + J_1(u_1 - \tilde{u}_1) + J_2(u_2 - \tilde{u}_2) \end{aligned} \quad (7)$$

By splitting J_1 in $\text{Diag } J_1 + \text{OffDiag } J_1$ in (7), we obtain:

$$y_1 - f_{1:r}(\tilde{u}) - J_2(u_2 - \tilde{u}_2) = (\text{Diag } J_1)(u_1 - \tilde{u}_1) + (\text{OffDiag } J_1)(u_1 - \tilde{u}_1)$$

$$\tilde{u}_1 + (\text{Diag}^{-1} J_1) (y_1 - f_{1:r}(\tilde{u}) - (\text{OffDiag } J_1)(u_1 - \tilde{u}_1) - J_2(u_2 - \tilde{u}_2)) = u_1$$

As $u \in \mathbf{u}$, $y_1 \in \mathbf{y}_1$, $\text{Diag}^{-1} J_1 \in \text{Diag}^{-1} \mathbf{J}_1$, $\text{OffDiag } J_1 \in \text{OffDiag } \mathbf{J}_1$, and $J_2 \in \mathbf{J}_2$, one gets

$$u_1 \in H_{f_{1:r}}((\mathbf{J}_1 \mathbf{J}_2), \tilde{u}, \mathbf{u}, \mathbf{y}_1) \quad (8)$$

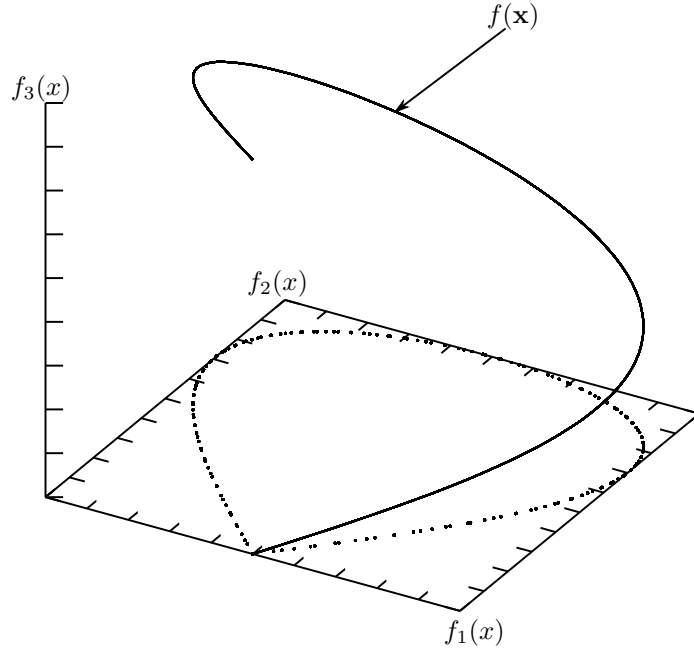


Figure 2: Example for $f : (\mathbf{x}) \subseteq \mathbb{R} \mapsto \mathbb{R}^3$ of constant rank 1.

and $u_1 \in \partial \mathbf{u}_1$ which contradicts (5). Then (5) implies (i).

(ii) By hypothesis, $f_{1:r}(\tilde{u}) \in \mathbf{y}_1$. \square

For a function $f : \mathcal{D} \subseteq \mathbb{R}^m \rightarrow \mathbb{R}^n$, Corollary 3.1 gives a test for a box to belong to the image of r components of f . It can only be performed if the Jacobian for r components of the function evaluated over the considered box is of full rank r . When the rank of f equals the dimension of the co-domain, f is a *submersion* [1], Corollary 3.1 can be used to compute an inner approximation of the entire range of f .

Example 3.2 Let $f : \mathcal{D} \subset \mathbb{R} \rightarrow \mathbb{R}^3$ be a function that satisfies (3) for $r = 1$.

$$f : \quad \mathbf{x} \subset \mathbb{R} \rightarrow \mathbb{R}^3$$

$$x \mapsto \begin{pmatrix} \sin 2x \\ \sin x \\ \frac{x}{2} \end{pmatrix} \quad (9)$$

The box $\mathbf{x} = [0, \pi] \subset \mathcal{D}$ is considered as the domain on which f is studied. The function is of constant rank 1 then using Corollary 3.1, one is able to compute an inner approximation of the range of a single component of f e.g. $f_1(\mathbf{x})$, $f_2(\mathbf{x})$ or $f_3(\mathbf{x})$. There is of course no proper box of dimension 2 or 3 included in the range of f . Figure 2 represents the range of the function defined in (9).

4 Algorithms

4.1 When domain and co-domain have the same dimension

Algorithm 1 in [5] computes an inner approximation for functions with domain and co-domain of the same dimension using Corollary 2.1 and a bisection algorithm. The method is as follows. For a given box $\tilde{\mathbf{x}}$ included in the initial domain \mathbf{x} , a box $\tilde{\mathbf{y}}$ such that $f(\mathbf{x}) = \{f(x) : x \in \tilde{\mathbf{x}}\} \subseteq \tilde{\mathbf{y}}$ is computed using the interval extension \mathbf{f} of f .

If the hypotheses of Corollary 2.1 are satisfied, $\tilde{\mathbf{y}}$ is part of an inner approximation of the range of f . If they are not satisfied, $\tilde{\mathbf{x}}$ is partitioned into two smaller boxes $\tilde{\mathbf{x}}'$ and $\tilde{\mathbf{x}}''$ that are treated like $\tilde{\mathbf{x}}$ was. If the box $\tilde{\mathbf{x}} = (\tilde{x}_1, \dots, \tilde{x}_m)$ is deemed too small to be further bisected (i.e. when $\text{wid}(\mathbf{x}) < \varepsilon$ where ε is a user-defined parameter), then the iterations stop for this box. This is described in Algorithm 1. It uses the function Inner described in Algorithm 2, to decide if a box belongs to the range of a function.

Algorithm 1: Range inner-approximation evaluation algorithm as defined in [5].

```

Input:  $f, \mathbf{x}, \epsilon$ 
Output:  $\mathcal{L}_{\text{Inside}}$  (list of boxes),  $\mathcal{L}_{\text{Boundary}}$  (list of boxes)
1  $\mathcal{L}_{\text{Inside}}$ : empty list of boxes;
2  $\mathcal{L}_{\text{Domain}}$ : empty list of boxes (sorted by decreasing width);
3  $\mathcal{L}_{\text{Domain}} \leftarrow \mathcal{L}_{\text{Domain}} \cup \{\mathbf{x}\}$ ;
4 while  $\mathcal{L}_{\text{Domain}}$  not empty do
5    $\tilde{\mathbf{x}} \leftarrow \text{Extract}(\mathcal{L}_{\text{Domain}})$ ;
6    $\tilde{\mathbf{y}} \leftarrow \mathbf{f}(\mathbf{x}) \cap (\mathbf{f}(\text{mid}(\tilde{\mathbf{x}})) + \mathbf{f}'(\tilde{\mathbf{x}})(\tilde{\mathbf{x}} - \text{mid}(\tilde{\mathbf{x}})))$ ; // Inclusion function
7   if Inner( $f, \mathbf{x}, \tilde{\mathbf{x}}, \tilde{\mathbf{y}}$ ) then
8     // Inner is from Algorithm 2 or Algorithm 3 according
      to the dimension of domain and co-domain
9      $\mathcal{L}_{\text{Inside}} \leftarrow \mathcal{L}_{\text{Inside}} \cup \{\tilde{\mathbf{y}}\}$ ;
10  else if  $\text{wid}(\tilde{\mathbf{x}}) \geq \epsilon$  then
11    Bisect the box  $\tilde{\mathbf{x}}$  to obtain  $\tilde{\mathbf{x}}'$  and  $\tilde{\mathbf{x}}''$ ;
12     $\mathcal{L}_{\text{Domain}} \leftarrow \mathcal{L}_{\text{Domain}} \cup \{\tilde{\mathbf{x}}', \tilde{\mathbf{x}}''\}$ ;
13  else
14     $\mathcal{L}_{\text{Boundary}} \leftarrow \mathcal{L}_{\text{Boundary}} \cup \{\tilde{\mathbf{y}}\}$ ;
15  end
16 end
17 return ( $\mathcal{L}_{\text{Inside}}, \mathcal{L}_{\text{Boundary}}$ );

```

Algorithm 2 decides for a given box $\tilde{\mathbf{x}} \subset \mathbf{x}$ whether $\mathbf{f}(\tilde{\mathbf{x}})$ belongs to the range of f over \mathbf{x} . The parameters τ and μ are used for the domain inflation (see Section 5.2 in [5]) and C is used to precondition the interval matrix \mathbf{J}^f (see Section 4 in [5]).

Algorithm 2: Inclusion test as defined in [5].

```

Input:  $f, \mathbf{x}, \tilde{\mathbf{x}}, \mathbf{y}$ 
Output: Boolean
1  $\tau \leftarrow 1.01; \mu \leftarrow 0.9;$ 
2  $k \leftarrow 0;$ 
3  $\tilde{x} \leftarrow \text{mid}(\tilde{\mathbf{x}});$ 
4  $C \leftarrow f'(\tilde{x})^{-1};$ 
5  $\mathbf{b} \leftarrow C\tilde{\mathbf{y}} - Cf(\tilde{x});$ 
6  $d_k \leftarrow +\infty; d_{k-1} \leftarrow +\infty;$ 
7 while  $d_k \leq \mu d_{k-1} \wedge \tilde{\mathbf{x}} \subseteq \mathbf{x}$  do
8    $\mathbf{u} \leftarrow \Gamma(Cf'(\tilde{\mathbf{x}}), \tilde{\mathbf{x}}, -\tilde{x}, \mathbf{b});$  //  $\Gamma$  is as defined in Corollary 2.1
9   if  $\tilde{x} + \mathbf{u} \subseteq \tilde{\mathbf{x}}$  then return true;
10   $d_{k-1} \leftarrow d_k;$ 
11   $d_k \leftarrow d(\tilde{\mathbf{x}}, \tilde{x} + \tau \mathbf{u});$  // Domain inflation (Cf. Section 5.2 in [5])
12   $\tilde{\mathbf{x}} \leftarrow \tilde{x} + \tau \mathbf{u};$ 
13 end
14 return false;

```

Example 4.1 Let $f(x) = Ax$ with $A = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}$, and an initial domain $\mathbf{x} = ([-2, 2], [-2, 2])$. The aim is to compute an inner approximation of the set $\{f(x) : x \in \mathbf{x}\}$. Of course, in this too simple case, direct methods would be applicable since A is an invertible matrix, but this is intended to exemplify the method. Figure 3 shows the result obtained using Algorithm 1. Since bisections occur in the domain, the result consists of a set of overlapping boxes, obtained by an inclusion function computing outer approximations. Dark areas in Figure 3 indicate many overlapping boxes.

4.2 When the domain and co-domain have different dimensions

We now extend the method in [5] to compute an inner approximation of the projection on r components of f . Algorithm 1 is used unchanged, except for the inner inclusion test Inner in Line 7 which is now implemented by Algorithm 3 instead of Algorithm 2. When using Algorithm 3, the vector-valued function f is assumed to satisfy (3). The main difference with the method in [5] is in the construction of the variables needed in the application of Corollary 3.1: in Algorithm 3, Lines 7–10 are dedicated to the definition of the vectors (u_1, u_2) , the interval vectors $(\mathbf{u}_1, \mathbf{u}_2)$ and the interval matrices $(\mathbf{J}^{f_{1:r}, u_1}, \mathbf{J}^{f_{1:r}, u_2})$ from Corollary 3.1. First, the r components must be separated from the others to obtain (u_1, u_2) . In Line 7, we construct from a vector in $\mathcal{D} \subseteq \mathbb{R}^m$, the initial

Algorithm 3: Inclusion test based on Corollary 3.1

Input: $f : \mathcal{D} \subseteq \mathbb{R}^m \rightarrow \mathbb{R}^n$, $\mathbf{x}, \mathbf{x}', \mathbf{y}, r$ // r is the rank of f
Output: Boolean

```

1  $\tau \leftarrow 1.01; \mu \leftarrow 0.9;$ 
2  $k \leftarrow 0;$ 
3  $\tilde{x} \leftarrow \text{mid}(\tilde{\mathbf{x}});$ 
4  $\mathbf{J} \leftarrow f'(\mathbf{x});$ 
5  $J^{f_{1:r}} \leftarrow \text{transformation}(f'(\tilde{x}), r);$  // recovery of the  $r$  first lines
   of the Jacobian of  $f$ 
6  $C \leftarrow \text{precondition}(J^{f_{1:r}});$  // Computation of the precondition
   matrix
7  $(\tilde{u}_1, \tilde{u}_2) \leftarrow \text{Extract}(\tilde{x}, r, m);$ 
8  $(\tilde{\mathbf{u}}_1, \tilde{\mathbf{u}}_2) \leftarrow \text{Extract}(\tilde{\mathbf{x}}, r, m);$ 
9  $(\mathbf{u}_1, \mathbf{u}_2) \leftarrow \text{Extract}(\mathbf{x}, r, m);$ 
10  $(\mathbf{J}^{f_{1:r}, u_1}, \mathbf{J}^{f_{1:r}, u_2}) \leftarrow \text{Extract}(C\mathbf{J}, r, m);$ 
11  $\mathbf{b} \leftarrow (C\tilde{\mathbf{y}} - Cf(\tilde{x}))_{1:r};$  // We need only the  $r$  first components
12  $d_k \leftarrow +\infty; d_{k-1} \leftarrow +\infty;$ 
13 while  $d_k \leq \mu d_{k-1} \wedge \tilde{\mathbf{u}} \subseteq \mathbf{u}$  do
14    $\mathbf{t} \leftarrow (\text{Diag}^{-1} \mathbf{J}^{f_{1:r}, u_1})(\mathbf{y}_1 - f_{1:r}(\tilde{u}_1, \tilde{u}_2) - (\text{OffDiag } \mathbf{J}_{f_{1:r}, u_1})(\tilde{\mathbf{u}}_1 - \tilde{u}_1) -$ 
      $\mathbf{J}^{f_{1:r}, u_2}(\tilde{\mathbf{u}}_2 - \tilde{u}_2));$ 
15   if  $\tilde{u} + \mathbf{t} \subseteq \tilde{\mathbf{u}}$  then return true;
16    $d_{k-1} \leftarrow d_k;$ 
17    $d_k \leftarrow d(\tilde{\mathbf{u}}, \tilde{u} + \tau \mathbf{t});$  // Domain inflation (Cf. Section 5.2 in
     [5])
18    $\tilde{\mathbf{u}} \leftarrow \tilde{u} + \tau \mathbf{t};$ 
19 end
20 return false;
```

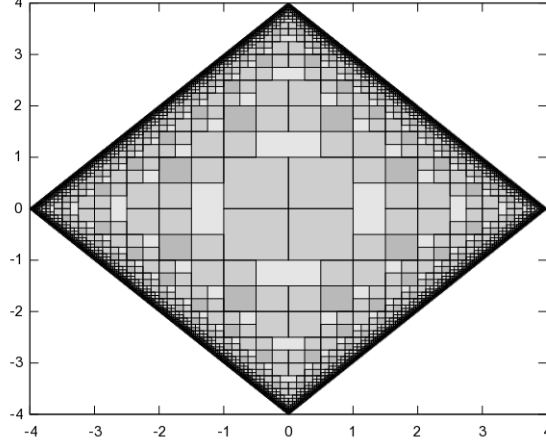


Figure 3: Inner approximation of the range of $f(x) = Ax$ when $x \in [-2, 2]^2$: $\mathcal{L}_{\text{Boundary}}$ (in black) and $\mathcal{L}_{\text{Inside}}$ are evaluated using Algorithm 1 with $\varepsilon = 10^{-3}$

domain, a vector in $\mathbb{R}^r \times \mathbb{R}^{m-r}$.

$$(x_1, \dots, x_m) \mapsto ((x_1, \dots, x_r), (x_{r+1}, \dots, x_m))$$

Lines 8 and 9 construct the same information as at Line 7 but for $\mathbf{x} \in \mathbb{R}^m$, an interval vector instead of a vector in \mathbb{R}^m , to get $(\mathbf{u}_1, \mathbf{u}_2) \in \mathbb{R}^r \times \mathbb{R}^{m-r}$. In Line 10, the pair of interval matrices $(\mathbf{J}^{f_{1:r}, u_1}, \mathbf{J}^{f_{1:r}, u_2}) \in \mathbb{R}^{r \times r} \times \mathbb{R}^{r \times m-r}$ are obtained from an interval matrix $\mathbf{J}^{f_{1:r}} \in \mathbb{R}^{r \times n}$.

Preconditioning In [5], the function has the same dimension for domain and co-domain and the Jacobian is then a square matrix. This interval square matrix which is an outer approximation of the Jacobian has to be preconditioned in order to apply the test in Corollary 2.1 with an H-matrix (see Definition 4.8 and Section 4 in [5]). Here we need also to extend this preconditioning operation. In practice, the preconditioning matrix is computed as follows: For a given box $\mathbf{u} \in \mathbb{R}^m$, $J^{f_{1:r}}(\tilde{\mathbf{u}})$, the Jacobian of the r first components of f , is computed for $\tilde{\mathbf{u}} = \text{mid}(\mathbf{u})$ and supplemented with the $(m-r)$ last lines of the identity matrix I_m to obtain an $m \times m$ matrix

$$D_{\tilde{\mathbf{u}}} = \begin{pmatrix} J^{f_{1:r}}(\tilde{\mathbf{u}}) \\ 0_{m-r \times r} \mid I_{m-r} \end{pmatrix}.$$

The inverse of $D_{\tilde{\mathbf{u}}}$ is computed and its r first columns are extracted to be the preconditioning matrix C . Decomposing C into $(C_1, C_2)^T$ with $C_1 \in \mathbb{R}^{r \times r}$ and $C_2 \in \mathbb{R}^{m-r \times r}$, the test in Corollary 3.1 becomes

$$H_{f_{1:r}}(C\mathbf{J}^{f_{1:r}}, \tilde{\mathbf{u}}, \mathbf{u}, C\mathbf{y}_1) \subseteq \text{int}(\mathbf{u}_1). \quad (10)$$

4.3 Extracting the sub-matrix of maximum rank from an interval matrix

The use of Algorithm 3 requires that the rank r of the Jacobian of f is known and that f satisfies (3). The Jacobian matrix is an interval matrix containing the Jacobian of f over some box. In the general case, we thus need to extract an interval sub-matrix of constant rank from the Jacobian of f .

In this section, we first define the rank of an interval matrix. Then, we propose different methods to extract sub-matrices of full rank from a given interval matrix. Some results on the evaluation of the eigenvalues of an interval matrix are well documented (see, e.g., [22]) but are not tractable for our problem. The extraction of an $r \times r$ sub-matrix of full rank is also not tractable. Thus, we chose to rely on three more tractable - though more approximate - methods aiming at extracting a sub-matrix of high rank from a given interval matrix.

Definition 4.2 (Regular interval square matrix [21])

Let $\mathbf{A} \in \mathbb{IR}^{n \times n}$ be an interval matrix. \mathbf{A} is regular if and only if for all matrix $A \in \mathbf{A}$, A is not singular.

Definition 4.3 (Rank of an interval matrix [9])

Let $\mathbf{A} \in \mathbb{IR}^{n \times m}$ be an interval matrix. \mathbf{A} is of constant rank r if and only if the largest regular interval square sub-matrix \mathbf{A}_0 of \mathbf{A} , is of dimension r .

Definition 4.3 means that for all $A \in \mathbf{A} \in \mathbb{IR}^{n \times m}$, the rank of A is larger than or equal to r . To extract a regular interval square matrix of dimension equal to the rank of \mathbf{A} , three techniques are proposed in what follows.

4.3.1 Building strictly dominant interval sub-matrices

This first method relies on the Levy-Desplanques theorem on strictly dominant matrices as a simple test for non-singularity. We uses this test to formulate the extraction of sub-matrices of full rank as a linear programming problem.

Definition 4.4 (Strictly diagonally dominant matrix [6])

Let $A = (a_{ij})_{1 \leq i, j \leq m} \in \mathbb{R}^{m \times m}$ be a square matrix. A is a strictly diagonally dominant matrix if and only if

$$\forall i \in \{1, \dots, m\}, |a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^m |a_{ij}|. \quad (11)$$

This definition can be extended to interval matrices, using magnitude and mignitude instead of the absolute value:

Definition 4.5 (Strictly diagonally dominant interval matrix [18])

Let $\mathbf{A} = (\mathbf{a}_{ij})_{1 \leq i, j \leq m} \in \mathbb{IR}^{m \times m}$ be a square interval matrix. \mathbf{A} is a strictly diagonally dominant interval matrix if and only if

$$\forall i \in \{1, \dots, m\}, \langle \mathbf{a}_{ii} \rangle > \sum_{\substack{j=1 \\ j \neq i}}^m |\mathbf{a}_{ij}| \quad (12)$$

	★	◆	◆	
	◆	◆	★	
	◆	★	◆	

Figure 4: Example of result provided by the method using strictly dominance (see Section 4.3.1) and the one using H-matrices (see Section 4.3.2) on an interval matrix $\mathbf{A} \in \mathbb{IR}^{4 \times 5}$: ★ represents components of the matrix that have been chosen ($x_{ij} = 1$) for the diagonal and ◆ represents the non-diagonal entries of the sub-matrix. Empty boxes represent components that are not part of the sub-matrix.

Theorem 4.6 (Levy-Desplanques theorem [14, 23]) *A strictly diagonally dominant (interval) matrix is regular.*

Consider $\mathbf{A} \in \mathbb{IR}^{n \times m}$, an interval matrix. We introduce the decision variables x_{ij} with $1 \leq i \leq n$ and $1 \leq j \leq m$. The boolean x_{ij} equals 1 if the component \mathbf{a}_{ij} of \mathbf{A} is picked to be an element of the diagonal of the rank k sub-matrix of \mathbf{A} and 0 otherwise. The x_{ij} s are obtained as solutions of the following constrained optimization problem

$$\begin{aligned} \max f(x) &= \sum_{i=1}^n \sum_{j=1}^m x_{ij} \\ \text{s.t.} \quad &\begin{cases} \sum_{i=1}^n x_{ij} \leq 1 & j = 1, \dots, m \\ \sum_{j=1}^m x_{ij} \leq 1 & i = 1, \dots, n \\ \sum_{k=1}^n x_{kl} \langle \mathbf{a}_{kl} \rangle > \sum_{i=1}^n \sum_{j=1}^m x_{ij} (1 - x_{il}) |\mathbf{a}_{il}| & \begin{matrix} k = 1, \dots, n \\ l = 1, \dots, m \end{matrix} \\ x_{ij} \in \{0, 1\} \end{cases} \end{aligned} \quad (13)$$

The objective is to maximize the size of a square regular sub-matrix of \mathbf{A} . The two first constraints ensure that at most one component on each row and column of the interval matrix \mathbf{A} is taken (it corresponds to the problem of placing towers in a possibly not square chess board). The last constraint corresponds to Theorem 4.6. Figure 4 shows an example of solution provided by the constrained optimization problem (13). A component of the interval matrix is picked if and only if it satisfies (12) and then leads to a strictly diagonally dominant interval matrix.

The last constraints in (13) are quadratic and have to be turned into linear constraints for efficiency reasons since linear programming techniques are generally fast. A given \mathbf{a}_{kl} is chosen if the sum of all the other \mathbf{a}_{il} for $i = 1, \dots, m$ for

which there exists an \mathbf{a}_{ij} that is part of the diagonal of the extracted sub-matrix is lower. Equivalently,

$$x_{kl} = 1 \Rightarrow \langle \mathbf{a}_{kl} \rangle > \sum_{\substack{i=1 \\ i \neq k}}^n \sum_{j=1}^m x_{ij} |\mathbf{a}_{il}|. \quad (14)$$

Using the so called Big-M relaxation (see, e.g., [8]), this constraint can be rewritten as follows.

$$\sum_{\substack{i=1 \\ i \neq k}}^n \sum_{j=1}^m x_{ij} |\mathbf{a}_{il}| \leq M + (\langle \mathbf{a}_{kl} \rangle - \mu - M)x_{kl} \quad (15)$$

with M chosen to be larger than $\sum_{\substack{i=1 \\ i \neq k}}^n \sum_{j=1}^m x_{ij} |\mathbf{a}_{il}|$ in order to deactivate the constraint when $x_{kl} = 0$ and μ as small as possible to approximate the strict inequality but not too small to avoid introduction of numerical instability. Using (15) in (13), the constrained optimization problem (13) becomes

$$\begin{aligned} \max f(x) &= \sum_{i=1}^n \sum_{j=1}^m x_{ij} \\ \text{s.t.} \quad &\begin{cases} \sum_{i=1}^n x_{ij} \leq 1 & j = 1, \dots, m \\ \sum_{j=1}^m x_{ij} \leq 1 & i = 1, \dots, n \\ \sum_{\substack{i=1 \\ i \neq k}}^n \sum_{j=1}^m x_{ij} |\mathbf{a}_{il}| \leq M + (\langle \mathbf{a}_{kl} \rangle + \mu - M)x_{kl} & \begin{matrix} k = 1, \dots, n \\ l = 1, \dots, m \end{matrix} \\ x_{ij} \in \{0, 1\} \end{cases} \end{aligned} \quad (16)$$

Using a linear programming solver on (16), a strictly dominant interval matrix can be extracted from \mathbf{A} . The property for an interval matrix \mathbf{A} to be a strictly dominant interval matrix is on the rows of \mathbf{A} . This definition can apply also for \mathbf{A}^T the transpose of \mathbf{A} , this is why the linear program is solved for both \mathbf{A} and \mathbf{A}^T to obtain the best result.

4.3.2 Building H-sub-matrices

A second method is now investigated. It uses a generalization of strictly dominant interval matrices, i.e., the notion of H-matrices [18]. Basic results on H-matrices are first provided before showing the slight changes in the constraint (15) that have to be done in order to detect H-sub-matrices in an interval matrix.

Definition 4.7 (Comparison Matrix [18])

Let $\mathbf{A} \in \mathbb{R}^{m \times m}$ be a square interval matrix. The comparison matrix $\langle \mathbf{A} \rangle$ is built

as follows

$$\langle \mathbf{A} \rangle_{ij} = \begin{cases} \langle \mathbf{A}_{ij} \rangle & \text{if } i = j \\ -|\mathbf{A}_{ij}| & \text{otherwise} \end{cases} \quad \text{with } i, j = 1, \dots, m.$$

Definition 4.8 (H-matrix [18])

Let $\mathbf{A} \in \mathbb{R}^{m \times m}$ be a square interval matrix. \mathbf{A} is an H-matrix if and only if there exists $u > 0_m$ such that $\langle \mathbf{A} \rangle u > 0_m$.

Theorem 4.9 ([18]) Every H-matrix is regular.

Remark 4.1 The notion of H-matrices generalizes the one of strictly dominant interval matrices since a strictly diagonally dominant interval matrix is a particular case of an H-matrix by fixing $u = \underbrace{(1, \dots, 1)^T}_m$ in Definition 4.8.

From Remark 4.1, only slight changes have to be done in order to detect an H-matrix instead of a strictly diagonally one. The constrained optimization Problem (16) is transformed into

$$\begin{aligned} \max f(x) &= \sum_{i=1}^n \sum_{j=1}^m x_{ij} \\ \text{s.t.} \quad &\begin{cases} \sum_{i=1}^n x_{ij} \leq 1 & j = 1, \dots, m \\ \sum_{j=1}^m x_{ij} \leq 1 & i = 1, \dots, n \\ \sum_{i=1}^n \sum_{j=1}^m x_{ij} |\mathbf{a}_{il}| u_{ij} \leq M + (\langle \mathbf{a}_{kl} \rangle u_{kl} - \mu - M) x_{kl} & k = 1, \dots, n \\ & l = 1, \dots, m \\ x_{ij} \in \{0, 1\} \\ u_{ij} > 0 \end{cases} \end{aligned} \tag{17}$$

Figure 4 shows an example of solution provided by the constrained optimization Problem (17). In (17), the last constraint requires a matrix of variables $U = (u_{ij})_{\substack{1 \leq i \leq n \\ 1 \leq j \leq m}}$ to be introduced. It corresponds to the vector u in Definition 4.8. This means we now have to solve a quadratic problem that could be tackled using SDP solvers. In order to solve (17) as efficiently as possible, i.e., by using linear programming techniques, we thus chose a particular u before solving (17). All components of u are chosen to be the inverse of the mignitude of the diagonal entries of the considered sub-matrix (as recommended in [18]):

$u = (\langle \mathbf{a}_{ij} \rangle^{-1})_{1 \leq i \leq n; 1 \leq j \leq m}$. The linear program is then

$$\begin{aligned} \max f(x) &= \sum_{i=1}^n \sum_{j=1}^m x_{ij} \\ \text{s.t.} \quad &\begin{cases} \sum_{i=1}^n x_{ij} \leq 1 & j = 1, \dots, m \\ \sum_{j=1}^m x_{ij} \leq 1 & i = 1, \dots, n \\ \sum_{\substack{i=1 \\ i \neq k}}^n \sum_{j=1}^m x_{ij} |\mathbf{a}_{il}| \langle \mathbf{a}_{ij} \rangle^{-1} \leq M + \underbrace{(\langle \mathbf{a}_{kl} \rangle \langle \mathbf{a}_{kl} \rangle^{-1})}_{=1} - \mu - M)x_{kl} & \begin{matrix} k = 1, \dots, n \\ l = 1, \dots, m \end{matrix} \\ x_{ij} \in \{0, 1\} \end{cases} \end{aligned} \quad (18)$$

Remark 4.2 Note that $\langle \mathbf{a}_{kl} \rangle$ has to be different from 0 because of the division that occurs in the constraint.

As in the method using strictly diagonally dominant matrices, the linear program (18) can be solved using a linear programming solver.

4.3.3 Combinatorial method

A random search for an interval sub-matrix of maximum rank is performed. It could rely on the two previous conditions (Definition 4.5 or Definition 4.8) to determine whether a matrix is of full rank. However, since no linear programming formulation has to be considered, one may use a more sophisticated test for full rank verification. We use for that a result provided in [21].

Theorem 4.10 (Corollary 5.1 in [21]) Let $\mathbf{A} \in \mathbb{IR}^{m \times m}$ be an square interval matrix. Let Δ be a matrix such that $\mathbf{A} = \text{mid}(\mathbf{A}) + [-\Delta, \Delta]$. Let $D = |\text{mid}(\mathbf{A})^{-1}| \Delta$. If the spectral radius $\rho(D) < 1$ then \mathbf{A} is regular.

We combine this criterion derived from Theorem 4.10 by extracting randomly chosen components of an interval matrix and testing whether the resulting sub-matrix is regular. This process is described in Algorithm 4.

4.3.4 Experiments on the sub-matrix extraction

In this section, some results on extracting an interval square sub-matrix of maximum rank from a given interval matrix are now described for the three methods that have been previously described. Two types of experiments have been performed depending on how the considered interval matrix has been produced. The linear programs for the first two methods have been solved using the GLPK interface for C++ [15]. All experiments have been done on a 2.3 Ghz Intel core i5 processor based laptop with 8 GBytes memory. In all experiments,

Algorithm 4: Extraction of regular interval sub-matrix

```

Input:  $\mathbf{A} \in \mathbb{R}^{n,m}$ 
Output:  $\mathbf{B} \in \mathbb{IR}^{k,k}$  a regular interval matrix
1 if  $n = m$  and  $\mathbf{A}$  is regular then
2   | return  $\mathbf{A}$ 
3 end
4 for  $k = \min(n, m)$  downto 1 do
5   | for  $i = 1$  to  $\text{MAX\_ITERATION}$  do
6   |   |  $\mathbf{B} \leftarrow \text{extraction}(\mathbf{A}, k)$  // Extraction takes randomly  $k$ 
        |   | components of  $\mathbf{A}$  to be in the diagonal of  $\mathbf{B}$  and the
        |   | other components are deducted from this diagonal.
7   |   | if  $\mathbf{B}$  is regular then
8   |   |   | return  $\mathbf{B}$ 
9   |   | end
10  | end
11 end

```

the constant for the linear programs (16) and (18) are $M = \sum_{i=1}^m \sum_{j=1}^n \bar{\mathbf{a}}_{ij}$ and $\mu = 10^{-2}$. For the random extraction, Algorithm 4 has been used with a fixed MAX_ITERATION equal to 500. Results have been averaged over 200 realizations.

First experiments have been done on an interval matrix generated randomly but containing a strictly dominant interval sub-matrix with a fixed dimension. The matrix is constructed as follows: the size m of an interval square matrix \mathbf{A} is chosen. For each component \mathbf{a}_{ij} of \mathbf{A} , $\mathbf{a}_{ij} = [\underline{\mathbf{a}}_{ij}, \bar{\mathbf{a}}_{ij}]$ with $1 \leq i, j \leq m$, $\underline{\mathbf{a}}_{ij}$ is a (pseudo) random number in $[0, 9]$ and $\bar{\mathbf{a}}_{ij}$ is equal to $\underline{\mathbf{a}}_{ij} + 1$ for all $i, j = 1, \dots, m$. An a priori rank r is chosen. Then r coordinates (i, j) are randomly picked and for each of these pairs, the associated interval $\mathbf{a}_{ij} = [\underline{\mathbf{a}}_{ij}, \bar{\mathbf{a}}_{ij}]$ is taken as

$$\underline{\mathbf{a}}_{ij} = 1 + \sum_{k=1}^m \bar{\mathbf{a}}_{kj} \text{ and } \bar{\mathbf{a}}_{ij} = \underline{\mathbf{a}}_{ij} + 1$$

Using this construction, there is in the resulting interval matrix \mathbf{A} an $r \times r$ interval sub-matrix of \mathbf{A} and r is a lower bound of the actual rank of \mathbf{A} .

Figure 5 depicts a first experiment showing the average execution time as a function of the dimension of the considered interval matrix for the three methods. The constructed matrices here are square, have a dimension from 2 to 8 and are strictly dominant interval matrices ($r = m$). This experiment shows the exponential increase of the computing time needed while the dimension of the initial matrix for the methods using an LP solver and the apparently better behaviour of the combinatorial method (with a fixed number of iterations). Figure 6 shows average computing times of the LP solver for the search of H-sub-matrices and strictly diagonally dominant sub-matrices, and for the random sub-matrix extraction for an interval square matrix of dimension 8. In this case

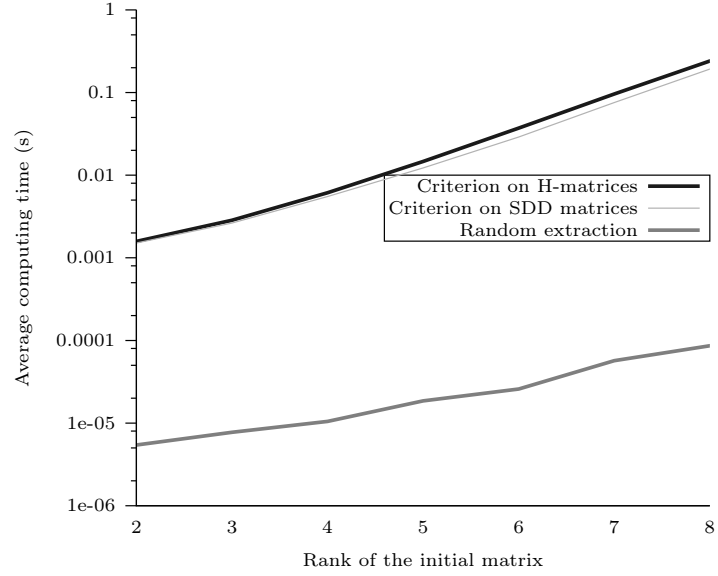


Figure 5: Computing time for the LP solver and the random extraction for a full rank interval matrix of size $i \times i$.

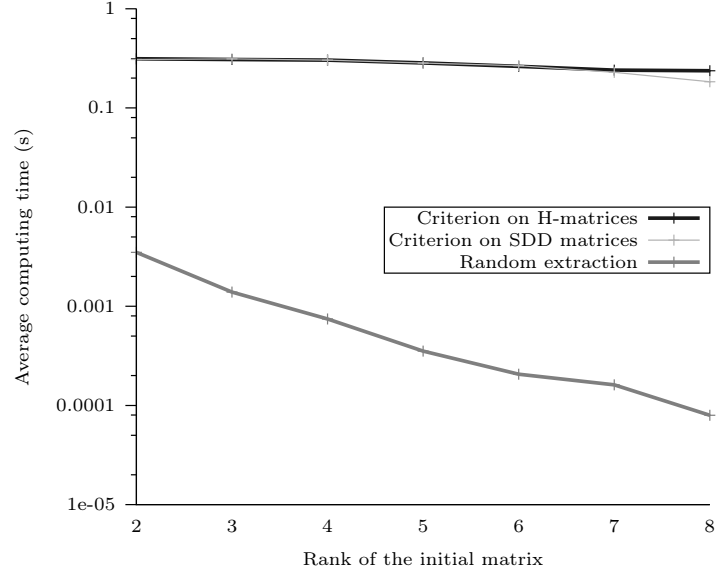


Figure 6: Average computing time (in seconds) over iterations of the LP solver for the search of H-sub-matrices and strictly diagonally dominant sub-matrices, and for the random sub-matrix extraction for an interval matrix of size 8×8 as a function of the minimum expected rank.

the matrix created randomly is constructed with a known minimum rank r with $2 \leq r \leq 8$ that is the size of the interval H sub-matrix created. Using the LP solver, the execution time is more or less constant. We stress here the fact that the method using random extraction becomes faster for larger ranks because they are detected very early with Algorithm 4 and because the matrix can contain components equal to 0, the method using H-matrices in the case where u is fixed as presented can no longer be applied since a division by 0 can occur in the linear program (18).

Finally two last experiments are provided: they use a different technique to construct the interval matrix. To test the efficiency of the proposed algorithms, matrices with known rank r are built. For that purpose, a triangular matrix A (with $(A)_{ii} \neq 0$, $1 \leq i \leq r$) is first created. It is the upper left part of a matrix $J \in \mathbb{R}^{n \times n}$ with $n \geq r$. The $(n - r)$ remaining columns of J are created as linear combinations of the r first columns. Then a sequence of rotations are applied to the matrix J (here we applied $n + 1$ rotations). An angle θ_i is chosen randomly with $0 \leq \theta_i \leq \pi$ to compose the rotation matrix. A pair (k, l) with $k \neq l$ and $1 \leq k, l \leq n$ is chosen randomly and the rotation is applied for coordinates (k, l) and (l, k) . The matrix A constructed in this way is not interval.

Figure 7 shows the results for a first experiment using this matrix construction which has the advantage to let us know the expected rank of the results. Using Theorem 4.6 as a criteria for sub-matrix extraction generally leads to a sub-matrix of dimension less than the rank of the matrix that is worse than the combinatorial method.

The matrix construction can be used to show the impact of the interval width of the components of the matrix on the method efficiency. For the last experiment, the same matrix construction than the previous one is used except that the components of the matrix constructed are thickened to obtain an interval matrix (the interval $[-0.25, 0.25]$ is added to each component of the matrix for one experiment and $[-0.5, 0.5]$ for another). Results are shown in Figure 8.

These experiments show that the combinatorial method is better than the method extracting strictly diagonally dominant sub-matrices. This is due to Theorem 4.10 which can detect a bigger subset of regular matrices than just strictly diagonally dominant ones.

Limitations

As previously mentioned, this method cannot guarantee to obtain the sub-matrix of maximum rank. For a given function $f : \mathcal{D} \subseteq \mathbb{R}^m \rightarrow \mathbb{R}^n$ of constant rank r , only $p \leq r$ components can be detected from the LP program (13). An inner approximation will be obtained only for these p components of f .

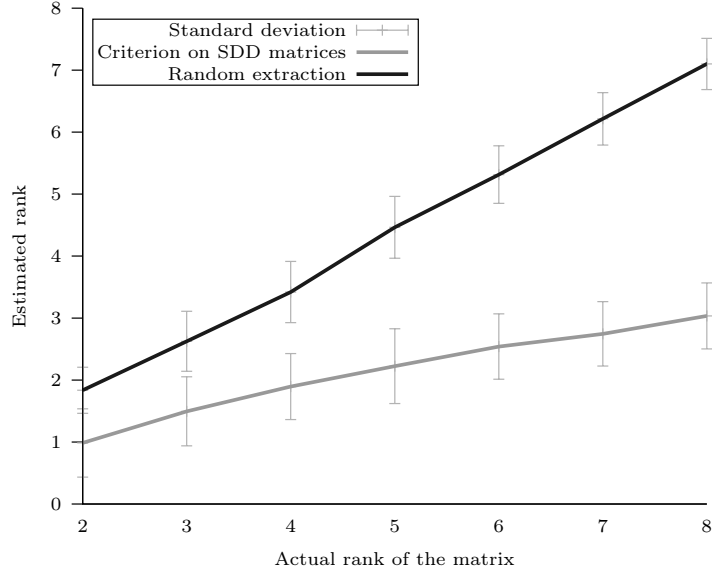


Figure 7: Execution from a matrix $J \in \mathbb{R}^{8 \times 8}$ of constant rank $i = 2, \dots, 8$. Results are average values over 200 computations.

5 Computation of an inner approximation of the range of a vector-valued function

This section shows some results of the computation of inner approximations of immersions and submersions. The functions considered in these examples satisfy (3) with r known.

5.1 Immersion

Consider the problem of finding the range of the function

$$f : \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

$$(u, v) \mapsto \begin{cases} f_1(u, v) = \cos(u) \cos(v) \\ f_2(u, v) = \sin(u) \cos(v) \\ f_3(u, v) = \sin(v) \end{cases} \quad (19)$$

over the box $(\mathbf{u}, \mathbf{v}) = ([\frac{3\pi}{2} + \tau, 2\pi - \tau]; [\tau, \frac{\pi}{2} - \tau])$, $\tau > 0$. f is of constant rank 2 in (\mathbf{u}, \mathbf{v}) . The rank is equal to the dimension of the domain of f , it is then an *immersion*. Corollary 3.1 can be used to get an inner approximation of the range of two components of f . Here we compute the range of the two first components, but the two last or the first and the last components could also be considered.

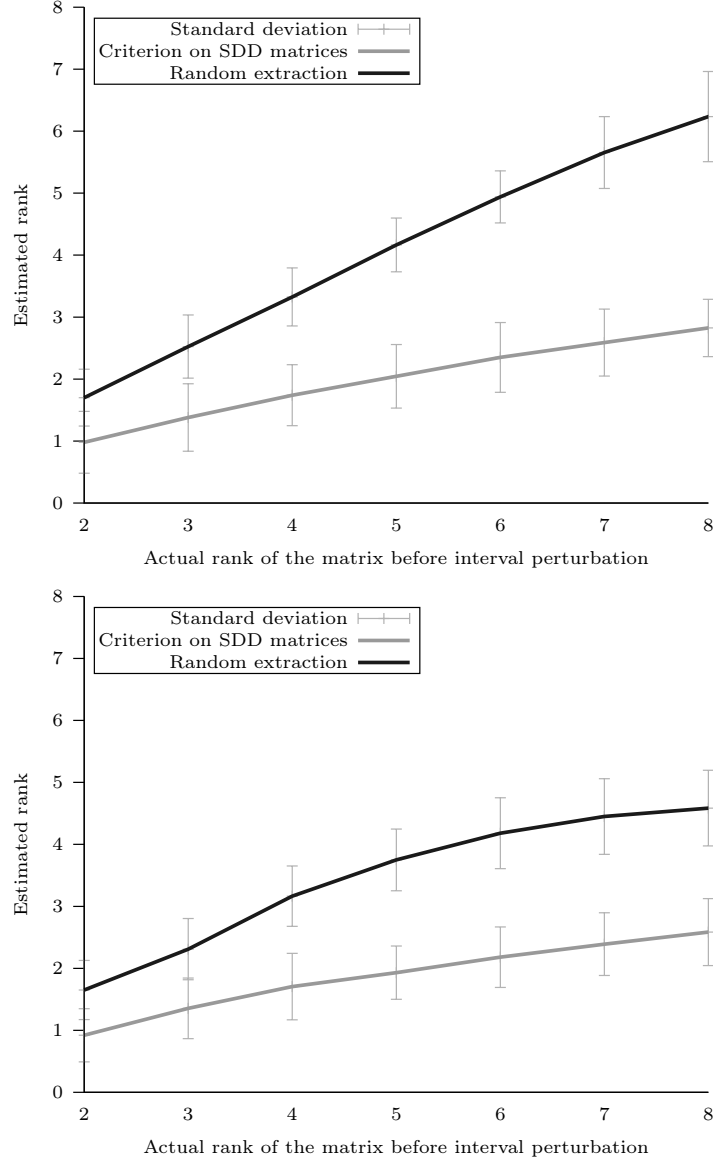


Figure 8: Estimate of the rank of an interval matrix $\mathbf{A} \in \mathbb{IR}^{8 \times 8}$ of constant rank $i = 1, \dots, 8$. Upper picture: $[-0.25, 0.25]$ is added to each component, lower picture: $[-0.5, 0.5]$ is added to each component. Results are average values over 200 evaluations. Standard deviation is provided.

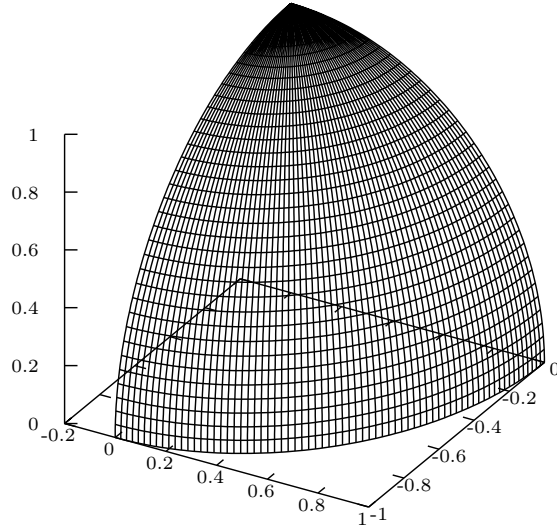


Figure 9: Example 1: Range of the immersion defined in (19) for the initial domain $(\mathbf{u}, \mathbf{v}) = ([\frac{3\pi}{2} + \tau, 2\pi - \tau]; [\tau, \frac{\pi}{2} - \tau])$, $\tau > 0$.

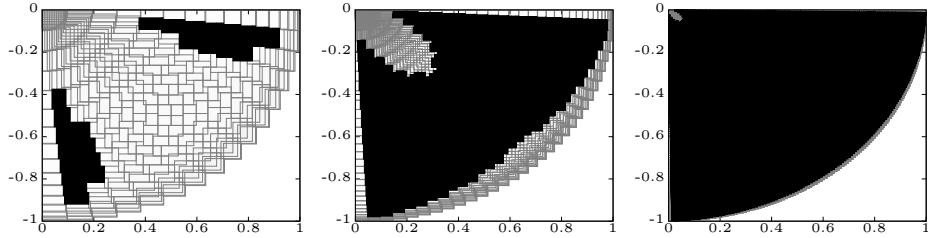


Figure 10: Example 1: Results of the computation of an inner approximation for different values of parameter ε (0.1, 0.06, and 0.02).

Figure 9 represents the image of (\mathbf{u}, \mathbf{v}) by f which has no volume in its co-domain \mathbb{R}^3 . Results of the computation of an inner approximation of this range are shown in Figure 10 for different values of ε in Algorithm 1. The smaller the ε , the more accurate the inner approximation and the longer the computing time. In Figure 10, empty boxes in gray represent boxes Algorithm 1 was unable to prove to be in the range. Black boxes all belong to the range.

The left part of Figure 10 is for $\varepsilon = 0.1$, the middle part for $\varepsilon = 0.06$ and the last part for $\varepsilon = 0.02$. The computing times are respectively 0.026 s, 0.10 s, and 0.64 s.

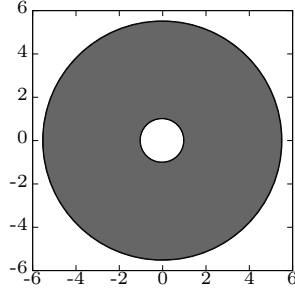


Figure 11: Example 2: Range of the submersion defined in (20) for the initial domain $(\mathbf{x}, \mathbf{y}, \mathbf{z}) = ([2, 4.5], [0, 2\pi - \tau], [0, 2\pi - \tau])$, $\tau = 10^{-3}$.

5.2 Submersion

Consider now the computation of an inner approximation of the range of the function

$$f : (\mathbf{x}, \mathbf{y}, \mathbf{z}) \subseteq \mathbb{R}^3 \rightarrow \mathbb{R}^2$$

$$(x, y, z) \mapsto \begin{cases} (x + r \cos(z)) \cos(y) \\ (x + r \sin(z)) \sin(y) \end{cases} \quad (20)$$

with $(\mathbf{x}, \mathbf{y}, \mathbf{z}) = ([2, 4.5], [0, 2\pi - \tau], [0, 2\pi - \tau])$, $\tau = 10^{-3}$. Figure 11 represents the range of f and Figure 12 represents different computations of an inner approximation according to the parameter ε in Algorithm 1. On Figure 12a $\varepsilon = 0.5$ and it took 0.18s to get the result. For Figure 12b, $\varepsilon = 0.3$ and computation time is 6.25s. In Figure 12c, it took 145.53s with $\varepsilon = 0.1$ to get these results. Finally Figure 12 d is for $\varepsilon = 0.05$ and the computing time is 838.67s. The time needed for computation is longer for this experiment than for the previous one on immersion. It is due to the fact that the Jacobian of f is not of full rank in the entire domain $([2, 4.5], [0, 2\pi - \tau], [0, 2\pi - \tau])$. It is why an area (cf. Figure 12d) remains out of the range of f .

6 Conclusion

Goldsztejn and Jaulin in [5] proposed a way to compute an inner approximation of the range of a vector-valued function. This paper provides an algorithm to evaluate inner approximations of the range of vector-valued functions without restriction on the dimension of its domain and co-domain. Using the proposed algorithm, one is able, for functions from \mathbb{R}^m to \mathbb{R}^n to evaluate an inner approximation of the projection of the range of f on at most r components, where r is the rank of the Jacobian matrix of f . In the general case, this rank r is unknown a priori, it is thus necessary to develop several techniques to extract a sub-matrix of maximal rank from a given interval matrix. The restriction of this method is providing an inner approximation of at most r components of the function if this function has a constant rank r .

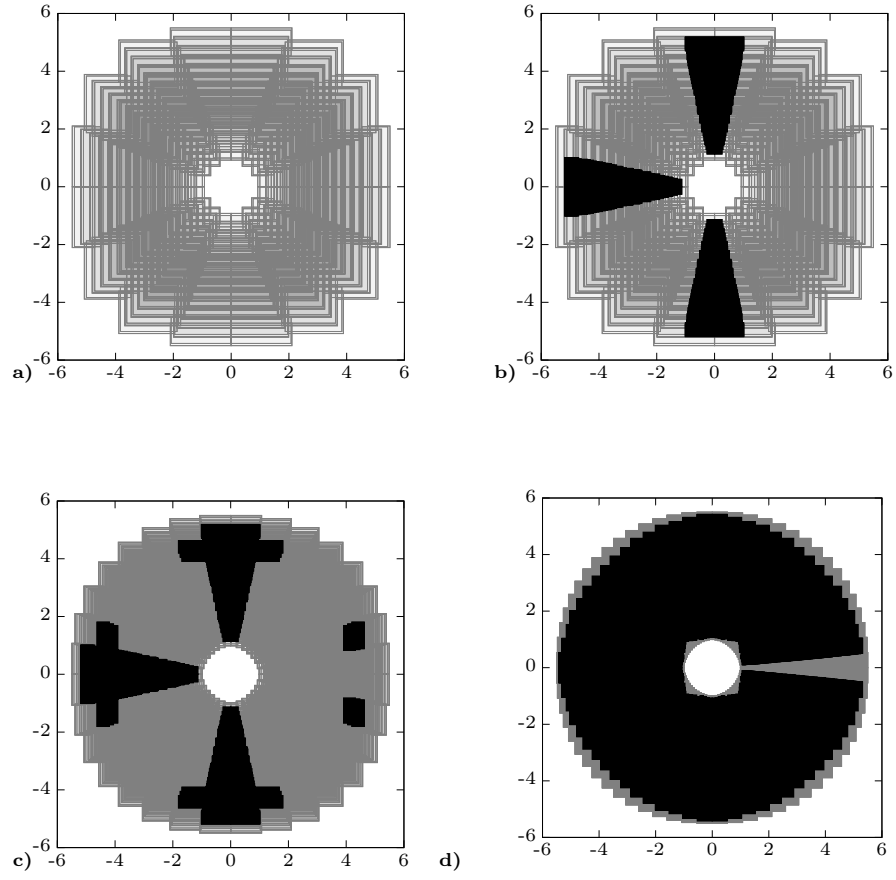


Figure 12: Example 2: Results of computation of an inner approximation according to the parameter ε in Algorithm 1 ($\varepsilon = 0.5, 0.3, 0.1$ and 0.05).

References

- [1] V.I. Arnold, S.M. Gusein-Zade, and Varchenko A.N. Singularities of differentiable maps, vol. 1. *Monographs in Mathematics*, 82, 1985.
- [2] J.P. Aubin and H. Frankowska. *Set-valued analysis*. Birkhäuser Boston, 2008.
- [3] P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fix-points. In *POPL '77: Proceedings of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pages 238–252. ACM Press, 1977.
- [4] A. Goldsztejn. A right-preconditioning process for the formal-algebraic approach to inner and outer estimation of ae-solution sets. *Reliable Computing*, 11(6):443–478, 2005.
- [5] A. Goldsztejn, L. Jaulin, et al. Inner approximation of the range of vector-valued functions. *Reliable Computing*, 14:1–23, 2010.
- [6] G.H. Golub and C.F. Van Loan. *Matrix computations*, volume 3. Johns Hopkins Univ Pr, 1996.
- [7] P. Herrero, M.Á. Sainz, J. Vehí, and L Jaulin. Quantified set inversion algorithm with applications to control. *Reliable Computing*, 11(5):369–382, 2005.
- [8] J.N. Hooker. *Integrated methods for optimization*, volume 170. Springer, 2011.
- [9] Y. Ishida and A. Nogi. Reasoning about structure of interval systems: An approach by sign directed-graph. In *Proceedings of the 10th International Workshop on Qualitative Reasoning (QR-96)*, 1996.
- [10] Daisuke Ishii, Alexandre Goldsztejn, and Christophe Jermann. Interval-based projection method for under-constrained numerical systems. *Constraints*, 17(4):432–460, 2012.
- [11] L. Jaulin, M. Kieffer, Olivier Didrit, and É. Walter. *Applied interval analysis: with examples in parameter and state estimation, robust control and robotics*. Springer Verlag, 2001.
- [12] L. Jaulin and É. Walter. Guaranteed tuning, with application to robust control and motion planning. *Automatica*, 32(8):1217–1221, 1996.
- [13] E. Kaucher. Interval analysis in the extended interval space IR. *Computing Suppl*, 2:33–49, 1980.
- [14] L. Levy. Sur la possibilité de l’équilibre électrique. *CR Acad. Sci. Paris*, 93:706–708, 1881.

- [15] A. Makhorin. GLPK (GNU linear programming kit). <http://www.gnu.org/software/glpk>, 2006.
- [16] R.E. Moore. *Interval analysis*, volume 60. Prentice-Hall Englewood Cliffs, New Jersey, 1966.
- [17] V.M. Nesterov. Interval and twin arithmetics. *Reliable Computing*, 3(4):369–380, 1997.
- [18] A. Neumaier. *Interval methods for systems of equations*, volume 37. Cambridge Univ Pr, 1990.
- [19] A. Neumaier. Complete search in continuous global optimization and constraint satisfaction. *Acta Numerica*, 13(1):271–369, 2004.
- [20] S. Poljak and J. Rohn. Checking robust nonsingularity is np-hard. *Mathematics of Control, Signals, and Systems (MCSS)*, 6(1):1–9, 1993.
- [21] J. Rohn. Systems of linear interval equations. *Linear algebra and its applications*, 126:39–78, 1989.
- [22] J. Rohn. Interval matrices: Singularity and real eigenvalues. *SIAM journal on matrix analysis and applications*, 14:82–82, 1993.
- [23] O. Taussky. A recurring theorem on determinants. *The American Mathematical Monthly*, 56(10):672–676, 1949.
- [24] L.W. Tu. *An introduction to manifolds*. Springer, 2010.